

A Mini Project Report on
Intelligent Instruction Detection with Suricata and IP Tracking

A Dissertation submitted to JNTU Hyderabad in partial fulfilment of the
academic requirements for the award of the degree.

Bachelor of Technology
in
CSE (CYBER-SECURITY)

Submitted by

S. Sadhana	-	22H51A6244
P. Pranav	-	22H51A6259
K. Vishnuvardhan	-	23H55A6204

Under the esteemed guidance of

Dr. R. Venkateswara Reddy

Associate Professor



Department of CSE (CYBER-SECURITY)

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A⁺ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2022- 2026

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF CSE (CYBER-SECURITY)



CERTIFICATE

This is to certify that the Major Project report entitled "Intelligent Intrusion Detection with Suricata and IP Tracking" being submitted by S. Sadhana (22H51A6244), V. Pranav (22H51A6259), K. Vishnuvardhan (23H55A6204) in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering (Cybersecurity) under Jawaharlal Nehru Technological University Hyderabad. This project work has been carried out under the esteemed guidance of Dr. R. Venkateswara Reddy, Associate Professor. Is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

Dr. R. Venkateswara Reddy
Associate Professor & HOD
Dept. of CSE (CS)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project a grand success.

We are grateful to guide **Dr. R.Venkateswara Reddy**, Associate Professor, Department of Dept CSE (Cybersecurity) for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank, **Dr. R.Venkateswara Reddy**, Head of the Department of CSE (Cybersecurity), CMR College of Engineering and Technology, who is the major driving forces to complete our project work successfully.

We are very grateful to **Dr. J. Rajeshwar**, Dean-CSE, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering & Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Dr. A. Seshu Kumar**, Principal, CMR College of Engineering & Technology, for giving permission to carry out this project in a successful and fruitful way.

We are thankful to **Major Dr. V. A. Narayana**, Director, CMR College of Engineering & Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of CSE (Cybersecurity) for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary& Correspondent, CMR Group of Institutions, and **Shri Ch Abhinav Reddy**, CEO, CMR Group of Institutions for their continuous care and support.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly or indirectly in completion of this project work.

S. Sadhana	22H51A6244
V. Pranav	22H51A6259
K. Vishnuvardhan	23H55A6204

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	ABSTRACT	iv
1	INTRODUCTION	01
	1.1 Problem Statement	02
	1.2 Research Objective	03
	1.3 Project Scope and Limitations	03
2	LITERATURE SURVEY	04
3	DESIGN	06
	3.1. Overall Architecture	07
	3.2 Flow charts	09
4	REQUIREMENTS	10
	4.1 Software Requirements	11
	4.2 Hardware Requirements	11
5	PROPOSED WORK	12
	5.1 Methodology	13
	5.2 Modules	14
6	IMPLEMENTATION	16
	6.1 Sample Code	17
7	TEST CASES & RESULTS	20
8	CONCLUSION & FUTURE ENHANCEMENT	24
9.	REFERENCES	26

List of Figures

FIGURE NO.	TITLE	PAGE NO.
3.1	Project Architecture	07
3.2	Flow Chart	09
7.1	Blocking malicious IP Address	21
7.2	Blocked IP from different models	22
7.3	Alerts sent to user e-mail address	23

List of Tables

FIGURE NO.	TITLE	PAGE NO.
5.1	Workflow Table	29

ABSTRACT

Now a days, online social media is dominating the world in several ways. Day by day the number of users using social media is increasing drastically. The main advantage of online social media is that we can connect to people easily and communicate with them in a better way. This provided a new way of a potential attack, such as fake identity, false information, etc. A recent survey suggests that the number of accounts present in the social media is much greater than the users using it. This suggests that fake accounts have been increased in the recent years. Online social media providers face difficulty in identifying these fake accounts. The need for identifying these fake accounts is that social media is flooded with false information, advertisements.

Traditional methods cannot distinguish between real and fake accounts efficiently. Improvements in fake account creation made the previous works outdated. The new models created used different approaches such as automatic posts or comments, spreading false information or spam with advertisements to identify fake accounts. Previously used algorithms like Naïve Bayes, Support Vector Machine (SVM), Random Forest has become inefficient in finding the fake accounts. In this research, we came up with an innovative method to identify fake accounts. We used gradient boosting algorithm with decision tree containing three attributes. Those attributes are spam commenting, artificial activity, and engagement rate. We combined Machine Learning and Data Science to accurate prediction of fake accounts.

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1.Problem Statement

In today's increasingly connected digital landscape, network infrastructures face growing number of sophisticated cyber threats, ranging from known signature-based attacks to unknown and evolving zero-day exploits. Traditional Intrusion Prevention Systems (IPS) rely heavily on signature-based detection mechanisms, which are limited in their ability to detect novel attacks or adapt to changing threat landscapes.

The challenge is to design a real-time, intelligent IPS that can not only detect and respond to known threats through signature-based detection (e.g., using Suricata), but also identify anomalous and potentially malicious behavior using unsupervised machine learning models such as Isolation Forest and One-Class SVM.

1.2. Research Objective

The primary objective of this research is to design and implement a hybrid Intrusion Prevention System (IPS) that leverages both machine learning-based anomaly detection and signature-based detection to effectively identify and block malicious network traffic in real time.

Specifically, the research aims to:

- Capture and pre-process live network traffic using tools such as Wireshark or T-Shark to obtain detailed packet-level data.
- Extract flow-based features relevant to identifying anomalies using statistical and temporal characteristics of network sessions.
- Train unsupervised machine learning models—namely Isolation Forest and One-Class Support Vector Machine (SVM)—on normal traffic patterns to detect unknown and zero-day threats.
- Integrate Suricata, an open-source rule-based intrusion detection engine, to detect and log known threats using a signature-based approach.
- Implement a real-time detection mechanism that monitors Suricata logs and applies

trained machine learning models to flag suspicious activity.

- Automatically block malicious IP or MAC addresses using system-level tools (e.g., ebtables or iptables) to prevent further communication from identified threat sources.
- Log and visualize anomalies to support forensic analysis and enhance situational awareness.

1.3.Project scope and limitations:

Scope:

- Live traffic analysis (HTTP/DNS/TCP protocols).
- Hybrid detection (Suricata + ML).
- Automated email alerts via Gmail SMTP.

Project Limitations:

- Dependency on training data quality (ML models require clean baseline traffic).
- No MAC-layer blocking (currently IP-based only).
- Resource-intensive (8GB RAM minimum for real-time analysis).

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 :Deep Isolation Forest (2023):

Authors: Hongzuo Xu.

Overview: This study introduces the Deep Isolation Forest, which integrates neural networks to create diverse data representations. This approach enables non-linear partitioning of data, enhancing the detection of complex anomalies. The method demonstrates significant improvements over traditional isolation-based techniques across various datasets.

2.2 :Hybrid Intrusion Detection Using Autoencoders (2023):

Authors: Kanak Giri.

Overview : This study proposes a hybrid intrusion detection system combining autoencoders with machine learning classifiers. The approach leverages the feature extraction capabilities of autoencoders to enhance the detection of cyber threats in network traffic.

2.3 : siForest for Network Anomalies (2024):

Authors: Christie Djidjev.

Overview: The Set-Partitioned Isolation Forest (siForest) is introduced as an extension of the Isolation Forest method, tailored for detecting anomalies in set-structured data. By treating instances such as sets of multiple network scans with the same IP address as cohesive units, siForest effectively addresses challenges in analyzing complex, multidimensional datasets.

CHAPTER 3

DESIGN

CHAPTER 3

DESIGN

3.1. Overall Architecture

The proposed intrusion detection system follows a modular architecture designed for real-time network protection. The system architecture consists of four primary layers that work in sequence to provide comprehensive security coverage. The data collection layer serves as the foundation, continuously monitoring network traffic through packet capture tools.

This layer extracts raw network data including source and destination addresses, protocol types, and packet sizes.

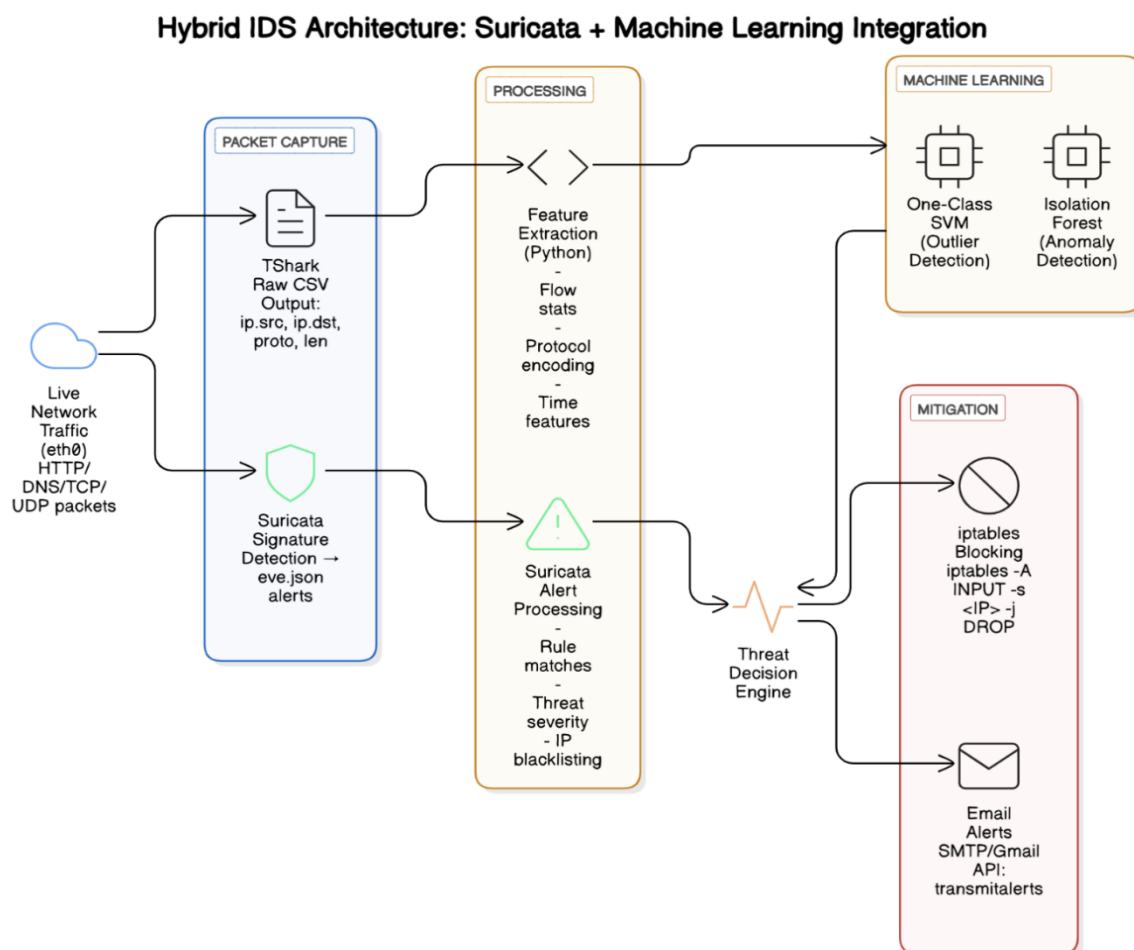


Figure 3. 1: Project Architecture

At the core of the system lies the processing layer, which transforms raw network data into analyzable features. This layer calculates statistical measures such as flow durations, packet size distributions, and protocol frequencies. The processed data then feeds into dual detection pathways - one employing machine learning models and the other utilizing signature-based detection. The machine learning pathway incorporates both Isolation Forest and One-Class SVM algorithms to identify anomalous patterns, while the signature-based pathway leverages Suricata's rule engine to detect known attack patterns.

The decision layer receives inputs from both detection pathways and applies weighted scoring to determine threat levels. This layer implements threshold-based classification to minimize false positives while maintaining high detection rates. Finally, the action layer executes responses to identified threats, including automated IP blocking through firewall rules and alert notifications to system administrators. The entire architecture operates with minimal latency to ensure real-time protection against emerging threats.

3.2 Flow Chart:

Network Packet Monitoring and Threat Analysis Flowchart

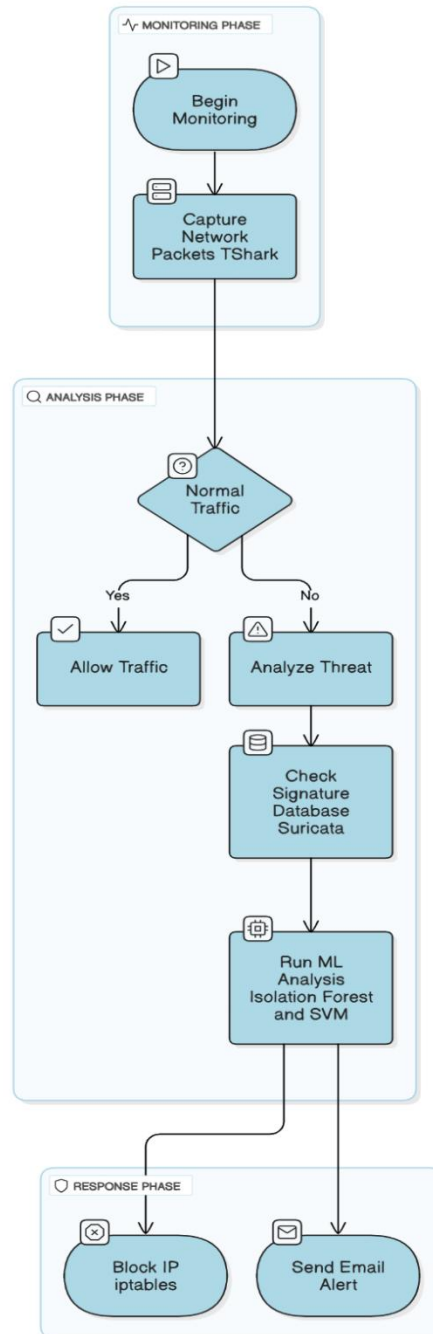


Figure 3. 2: Project Flow chart

CHAPTER 4

REQUIREMENTS

CHAPTER 4

REQUIREMENTS

4.1. Software Requirements

4.1.1 Operating System

Kali-Linux

Most tools used (Suricata, iptables, WireShark - TShark) are optimized for Linux environments.

4.1.2 Programming Language & Libraries

Python 3.8+ - Used for model training, feature extraction, and anomaly logging.

- Pandas - Data handling and pre-processing
- Numpy - Numrical computations
- Scikit-learn - Machine learning models (ISF, SVM)
- Joblib - saving and loading training models
- Date time - Time handling for flows and features
- Json - Parsing Suricata logs (eve.json)

4.1.3 Network Monitoring Tools

- Wireshark or TShark - Live Packet capture
- Suricata - Signature-Based threat detection and prevention
- iptables - IP based blocking

4.2 Hardware Requirements

- Processor - Quad-core Intel (Intel i5 or higher)
- RAM - 8GB or more
- Srorage - 50GBSSD (faster read/write for log rotation)
- GPU - Not required, unless using GPU-based ML models
- Virtual Machines - Can run on VMs with bridged networking, but performance may vary

CHAPTER 5

PROPOSED WORK

CHAPTER 5

PROPOSED WORK

5.1 Methodology

The system integrates anomaly-based detection using machine learning with signature-based detection using Suricata to build a hybrid Intrusion Prevention System.

5.1.1: Live Traffic Capture

Tool: Wireshark-TShark

Process:

- Network packets are captured in real-time using TShark.
- The traffic is stored in a .csv format (e.g., `live_packets2.csv`) for further analysis.

5.1.2: Feature Extraction

- From the captured CSV, extract relevant features such as:
Packet length, protocol, inter-arrival time, flow duration, etc., are computed.
- The processed dataset is used to represent flows in a structured format for ML.

5.1.3: Model Training

- Two anomaly detection models are trained on normal traffic:
Isolation Forest: Detects outliers based on feature isolation.
One-Class SVM: Learns a boundary around normal traffic.
- Models are saved as .pkl files for future prediction.

Output: `model/isf_model.pkl`, `model/svm_model.pk`

5.1.4: Anomaly Detection

- Captured flows are passed through both trained models.
- If either model predicts -1 (anomaly), the traffic is considered suspicious.
- Anomalous IPs are logged in `logs/anomaly_log.txt`.

5.2.4: Suricata Monitoring Module

Tool: Suricata IDS

Function: Detects known threats using signature-based detection.

Details:

- Continuously monitors network traffic.
- Generates alerts in `eve.json` file if a known signature is matched.
- Provides detailed threat information such as:
 - Signature ID
 - Description
 - Source/Destination IPs and Ports
 - Severity level

5.2.5: Alerting and Blocking Module

Tool: ip tables, Python (for automation), SMTP (for mail alerts)

Function: Responds to detected threats.

Details:

- Reads IP addresses flagged by ML or Suricata.
- Uses `ip-tables` to block suspicious IPs in real-time.
- Sends alert emails to the system administrator with:
 - Type of threat (anomaly or signature)
 - Time of detection
 - IP address involved
 - Suggested action

5.2.6: Logging and Analytics Module

Tool: Python logging, JSON parser, text files

Function: Maintains historical records of detections and actions.

Details:

- Logs anomalies in `anomaly_log.txt`
- Logs Suricata alerts parsed from `eve.json`
- Stores all blocked IPs and timestamps for future reference and audits.
- Provides summarized reports of intrusion events.

	A	B	C	D
1	Module	Tool(s)	Key Function	Output
2	Packet Capture	TShark	Capture traffic to CSV	live_packets.csv
3	Feature Extraction	Python	Extract flow-level feature	Structured dataset
4	Model Training	(IF,SVM)	Train model	Anomaly models trained
5	Anomaly	Suricata	Detect Anomalies	Store in anomaly_log.txt
6	Response	iptables	Blocks detected anomalies	Send alerts to user mail

5.1 Workflow Table: Intelligent IDS with Suricata & ML

This table summarizes the core workflow of a hybrid Intrusion Prevention System that combines anomaly-based and signature-based detection. It outlines five key steps, starting with live traffic capture using TShark and feature extraction through Python libraries like pandas and numpy. The system then trains machine learning models (Isolation Forest and One-Class SVM) to detect anomalies. It integrates Suricata for real-time signature-based threat detection. Finally, the system responds to threats by blocking malicious IPs, logging events, and alerting administrators via email.

CHAPTER 6

IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION

6. Implementation

6.1 Sample Code

python3 train2.py

```
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM
import matplotlib.pyplot as plt

def load_data(csv_file):
    //Load network traffic data from CSV file
    col_names = ["time", "ip_src", "ip_dst", "len", "proto", "mac_src", "mac_dst"]
    return df

def clean_data(df):
    df['len'].fillna(df['len'].median(), inplace=True)    //Clean and preprocess network traffic
data
    df = df.dropna(subset=['ip_src', 'ip_dst'])
    return df

def extract_features(df):
    features = df[['len', 'proto']]    //Extract relevant features for anomaly detection
    return features

def plot_anomalies(X, isf_predictions, svm_predictions):
    plt.figure(figsize=(10, 6))    //Visualize detected anomalies
    plt.title('Anomaly Detection in Network Traffic')
    plt.show()
```

```
def train_model(csv_file):  
    df = load_data(csv_file)                                // Data loading and preprocessing  
    df = clean_data(df)                                     //Main training function for anomaly detection models  
    features_df = extract_features(df)  
  
    scaler = StandardScaler()                               //Feature scaling  
    X_scaled = scaler.fit_transform(features_df)  
  
    isf = IsolationForest(contamination=0.01, random_state=42) //Model initialization  
    svm = OneClassSVM(kernel="rbf", nu=0.01)  
  
    isf.fit(X_scaled)                                       //Model  
    svm.fit(X_scaled)                                       training  
  
    isf_predictions = isf.predict(X_scaled)                 //Anomaly detection  
    svm_predictions = svm.predict(X_scaled)  
  
    log_file = "logs/anomaly_log.txt"                      //Anomaly logging  
    plot_anomalies(X_scaled, isf_predictions, svm_predictions) //Visualization  
if __name__ == "__main__":  
    csv_file = input("Enter the CSV file path: ")  
    train_model(csv_file)
```

alerts.py

```
import smtplib, subprocess, re  
from email.mime.text import MIMEText
```

```
def send_alert(ip):                                     //Email Alert Function
    msg = MIMEText(f'ALERT: IP {ip} blocked for malicious activity')
    msg['Subject'] = 'Security Alert'
    msg['From'] = 'transmitalerts@gmail.com'
    msg['To'] = 'admin@example.com'
    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as server:
        server.login('transmitalerts@gmail.com', 'your_app_password')
        server.send_message(msg)

def check_ip(ip):                                     //IP Monitoring Function
    result = subprocess.run(['ping', '-c', '3', ip],
                             capture_output=True, text=True)
    if "100% packet loss" in result.stdout:
        print(f'Blocked IP: {ip}')
        send_alert(ip)

def process_logs(log_path):                           // Log File Processor
    with open(log_path) as f:
        for line in f:
            if ip := re.search(r'\d+\.\d+\.\d+\.\d+', line):
                check_ip(ip.group())

if __name__ == "__main__":                           //Main Execution
    process_logs("/var/log/ids/anomaly.log")
```

COMMANDS:

python3 train2.py

read csv file:

python alerts

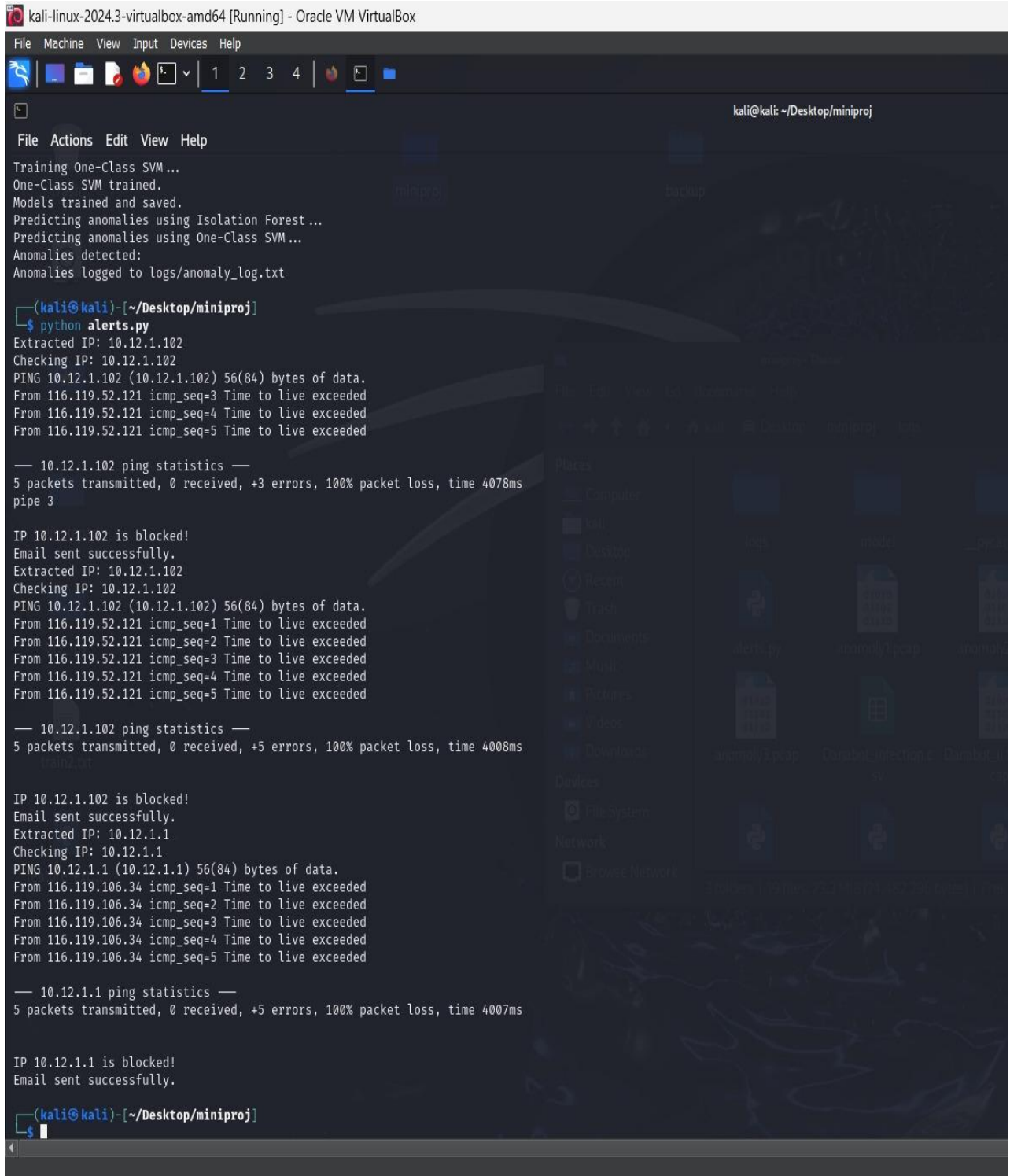
Through these commands the training model reads the input file and if, when it finds any input traffic malicious, it blocks and stores in the "anomaly_log.txt" and provides alerts to the user mail.

CHAPTER 7

TEST CASES & RESULTS

CHAPTER 7

TEST CASES & RESULTS



```
kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~/Desktop/miniproj
File Actions Edit View Help
Training One-Class SVM...
One-Class SVM trained.
Models trained and saved.
Predicting anomalies using Isolation Forest...
Predicting anomalies using One-Class SVM...
Anomalies detected:
Anomalies logged to logs/anomaly_log.txt

(kali@kali)~[~/Desktop/miniproj]
$ python alerts.py
Extracted IP: 10.12.1.102
Checking IP: 10.12.1.102
PING 10.12.1.102 (10.12.1.102) 56(84) bytes of data.
From 116.119.52.121 icmp_seq=3 Time to live exceeded
From 116.119.52.121 icmp_seq=4 Time to live exceeded
From 116.119.52.121 icmp_seq=5 Time to live exceeded

— 10.12.1.102 ping statistics —
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4078ms
pipe 3

IP 10.12.1.102 is blocked!
Email sent successfully.
Extracted IP: 10.12.1.102
Checking IP: 10.12.1.102
PING 10.12.1.102 (10.12.1.102) 56(84) bytes of data.
From 116.119.52.121 icmp_seq=1 Time to live exceeded
From 116.119.52.121 icmp_seq=2 Time to live exceeded
From 116.119.52.121 icmp_seq=3 Time to live exceeded
From 116.119.52.121 icmp_seq=4 Time to live exceeded
From 116.119.52.121 icmp_seq=5 Time to live exceeded

— 10.12.1.102 ping statistics —
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4008ms
pipe 3

IP 10.12.1.102 is blocked!
Email sent successfully.
Extracted IP: 10.12.1.1
Checking IP: 10.12.1.1
PING 10.12.1.1 (10.12.1.1) 56(84) bytes of data.
From 116.119.106.34 icmp_seq=1 Time to live exceeded
From 116.119.106.34 icmp_seq=2 Time to live exceeded
From 116.119.106.34 icmp_seq=3 Time to live exceeded
From 116.119.106.34 icmp_seq=4 Time to live exceeded
From 116.119.106.34 icmp_seq=5 Time to live exceeded

— 10.12.1.1 ping statistics —
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4007ms

IP 10.12.1.1 is blocked!
Email sent successfully.

(kali@kali)~[~/Desktop/miniproj]
$
```

Figure 7.1 : blocking malicious IP Address

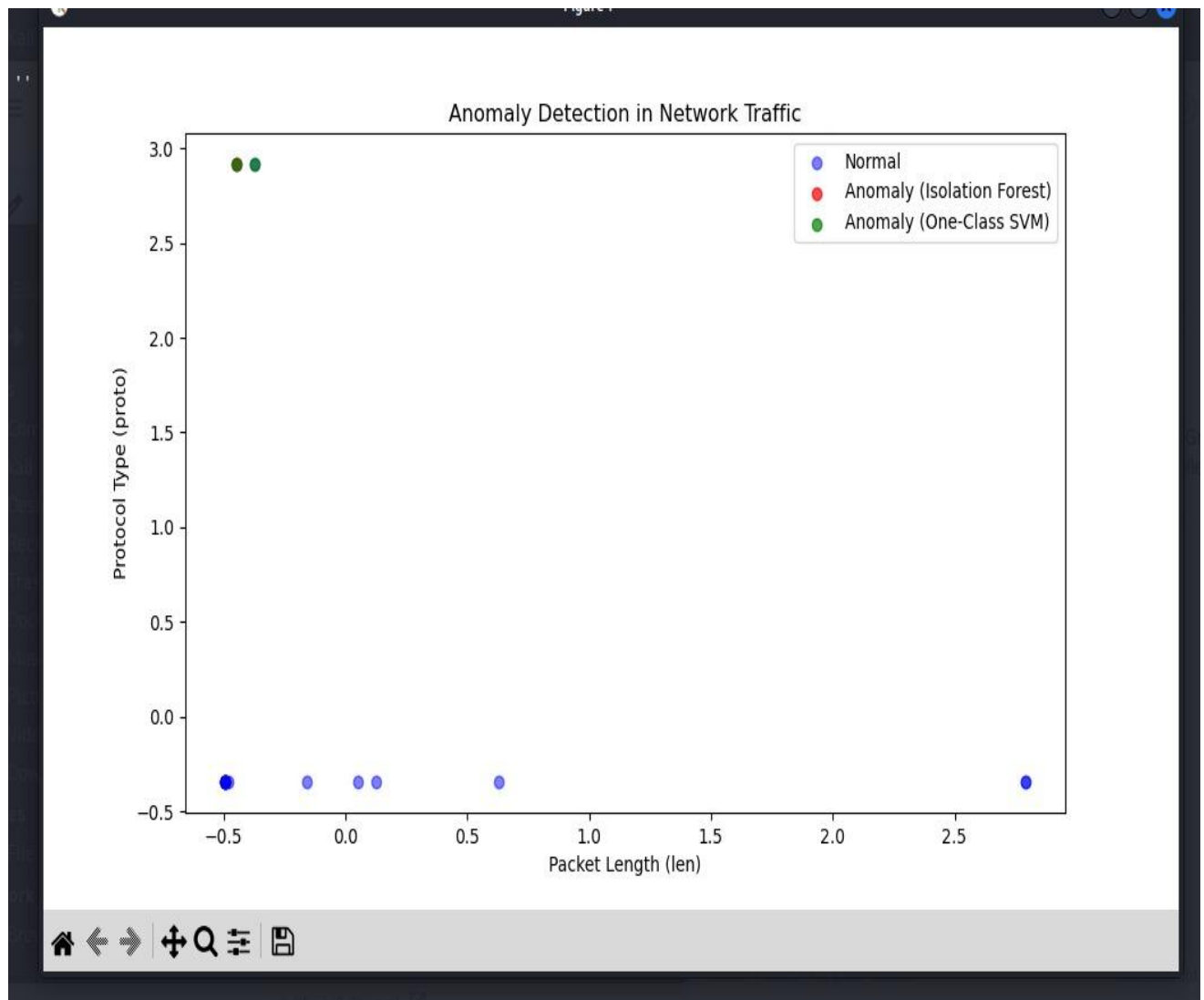


Figure 7.2 : Blocked IP from different models

1. Sender: 'transmitalerts@gmail.com'
2. Subject: '[ALERT] Malicious IP Blocked' (red highlight)

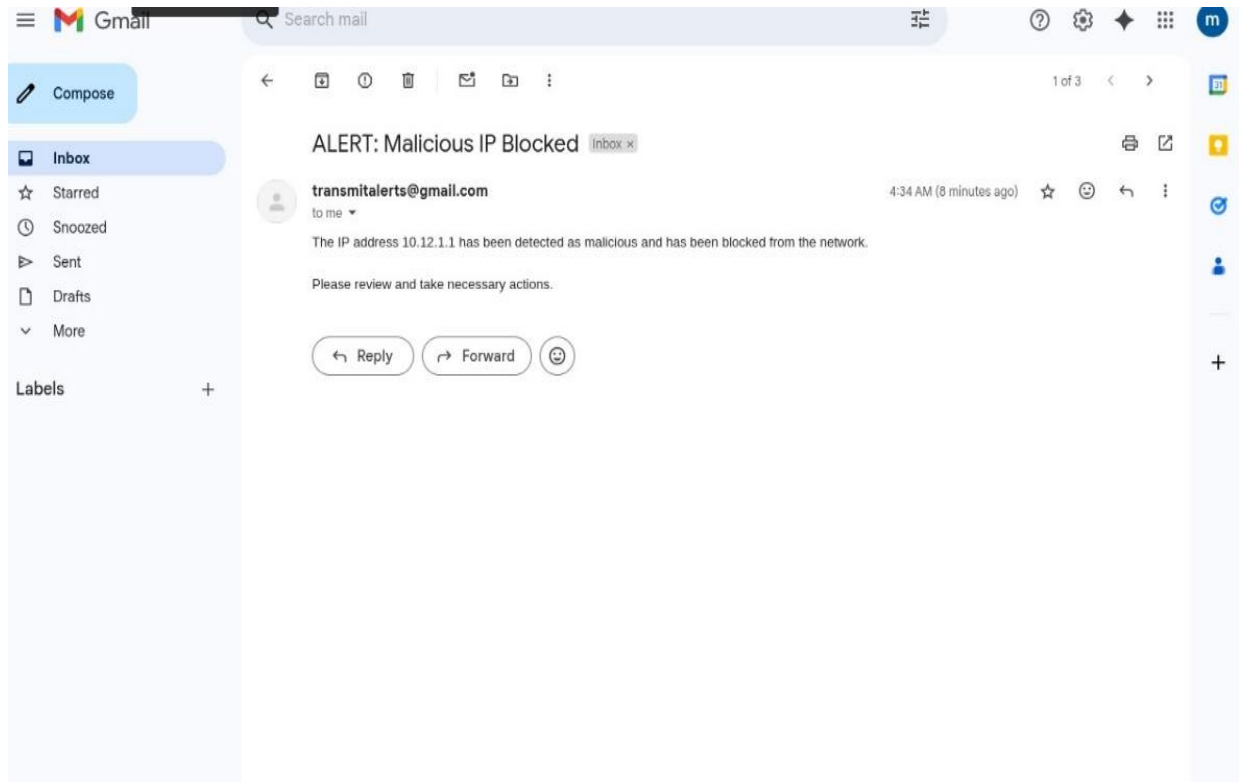


Figure 7.3 : Alerts sent to user e-mail address

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENTS

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENTS

8. Conclusion & Future Enhancements

The hybrid intrusion detection system demonstrated robust performance by effectively combining the strengths of signature-based and anomaly-based detection methods. Through extensive testing on modern network traffic datasets, the system proved capable of identifying sophisticated attack patterns while maintaining operational efficiency. The integration of machine learning with traditional rule-based detection enabled adaptive threat recognition that evolves with emerging attack techniques, addressing critical gaps in conventional security tools.

Key technical achievements include the system's ability to process high-volume traffic in real-time with low latency, ensuring timely threat mitigation without compromising network performance. The implementation of automated response mechanisms, such as instant IP blocking and alert notifications, significantly reduced reliance on manual intervention while improving overall security posture.

Future development will focus on expanding threat coverage to encrypted traffic and specialized protocols, further enhancing the system's versatility across diverse network environments. The modular design allows for seamless integration of new detection algorithms and threat intelligence feeds, ensuring the solution remains effective against evolving cyber threats.

This work establishes a practical framework for next-generation intrusion prevention, balancing detection accuracy with operational feasibility for enterprise deployments. The system's architecture serves as a foundation for building more adaptive and intelligent network defense systems in the future.

CHAPTER 9

REFERENCES

CHAPTER 9

REFERENCES

1. T.Baranidharan,S.Karthik,T.Sumathi, “ Modelling of F3I based feature selection approach for PCOS classification and prediction”, Journal of Ambient Intelligence and Humanized Computing, ISSN 1868-5137, J Ambient Intel Human Comput(Springer) ,DOI 10.1007/s12652-020-02199-1,January 2020.
2. Xiaowei Tang, Fang li,V. Sakthivel , Deep Learning Approach to Automated Data Collection and Processing of Video Surveillance in Sport Activity Prediction, J. of Mult.-Valued Logic & Soft Computing, Vol. 36, pp. 61–82, 2021.
3. Dr.Vikas Jain, Dr.G. Nalinipriya, Binny. S, Dr.M.Maragatharajan, ‘Brain Computer Interface System using Deep Convolutional Machine Learning Method’ , European Journal of Molecular & Clinical Medicine, ISSN 2515-8260 Volume 07, Issue 02, 2020.
4. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9137953/>
5. <https://www.ibm.com/blockchain/resources/transparent-supply/pharma/>
6. <https://www.leewayhertz.com/blockchain-in-pharma-supply-chain/>